УДК 681.326 **doi:** https://doi.org/10.20998/2522-9052.2022.3.02

Г. В. Хаханова

Харківський національний університет радіоелектроніки, Харків, Україна

ВЕКТОРНИЙ МЕТОД ПОШУКУ ПОСЛІДОВНОСТЕЙ У ВЕЛИКИХ ДАНИХ

Анотація. Пропонується технологічне програмне рішення для метричного пошуку та ідентифікації логіко-часових патернів бізнес-потоку даних за рахунок створення додаткових векторних структур даних та паралельного методу їх обробки. **Предметом досліджень** є методи пошуку та ідентифікації логіко-часових патернів у великих даних. **Метою** є підвищення ефективності пошуку та розпізнавання логіко-часових патернів, що семантично утворюють бізнес-функціональності у 8-годинному часовому фреймі скріншотів зі «сміттєвими» даними. **Застосовувані методи**: апарат теорії множин та булевої алгебри, метричні матричні моделі визначення параметрів для множин двійкових векторів, елементи теорії ймовірностей, теорія алгоритмів, програмне моделювання, аналіз великих даних. *Отримані результати*: метод пошуку та розпізнавання патернів на основі векторного завдання символьних послідовностей, які ідентифікують патерни у потоках великих даних, що використовує унітарне кодування інформаційних примітивів та даних; векторні моделі – структури унітарно-кодованих даних для опису потоку великих даних, як декартові добутки множини примітивів-string-маркерів та дискретної послідовності-реалізації заданого часового фрейма. **Практична значущість** роботи полягає у реалізації векторного методу, що дозволило створити програму розпізнавання патернів у потоці великих даних з ймовірністю 0,77%.

Ключові слова: логіко-часові патерни; бізнес-функціональності; часовий фрейм; пошук та ідентифікація; метрика перетину-об'єднання; верифікація коду.

1. Постановка проблеми

Детермінований цифровий комп'ютер (DC) завжди краще за будь-яку імовірнісну технологію, будь то: машинне навчання, нечітка логіка, штучний інтелект, нейромережі, еволюційні алгоритми. Всі AI-технології мають на меті досягти у своїй досконалості або навчанні детермінізму класичного комп'ютингу. Метрикою для досягнення такої досконалості є критичні параметри: Yield – якість обчислювача або розпізнавання та Time-to-Market – час для досягнення такої якості. Інтерес представляє співвідношення між AI і класичним комп'ютингом у метриці точності (якості) розв'язання задачі та часу Yield – Time-to-Market створення придатного для використання на практиці обчислювача, рис. 1.

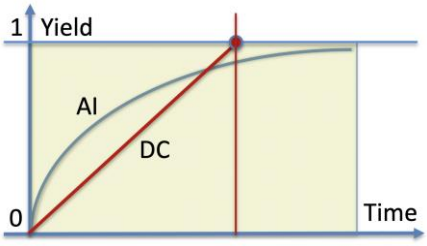

Рис. 1. Метричне відношення між AI- та детермінованим комп'ютером (**Fig. 1.** Metric relation between AI- and deterministic computer)

Таким чином, DC характеризується як інтелектуальна творчість (human intelligence), коли потрібно інтелектуальне напруження, щоб отримати точний розв'язок, а AI – є рутинне навчання (routine training), що призведе до ймовірнісного неточного розв'язку за тривалий час. Іншими словами, штучний інтелект не вимагає природного інтелекту, а детермінований комп'ютинг обов'язково використовує високий рівень природного інтелекту. Виходить, що дедуктивні методи, які мають апріорну точність логічних виразів або аналітичних формул, є кращими для їх імплементації в практику, ніж індуктивні методи, що вимагають тривалого навчання та верифікації для впровадження у виробничі процеси. Але залишається велике поле невизначених (неповних, нечітких) знань про процес чи явище, де тимчасовим виходом может служити ймовірнісний AI-розв'язок задачі.

2. Аналіз останніх досліджень і публікацій

Метрично визначити належність задачі до потенційного детермінізму чи ймовірнісного підходу її вирішення – було, є і буде моментом істини у творчості на полі emerging computing. Деяким підтвердженням сказаного є трендова картина топ-технологій від компанії Gartner [1], представлена на рис. 2.

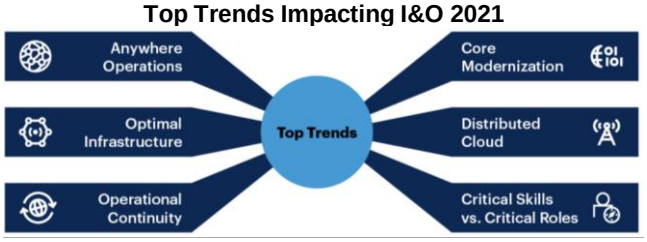

Рис. 2. Напрямки впливу моделей інфраструктури та операцій на бізнес (**Fig. 2.** Directions of influence of infrastructure and operations models on business)

Тут наголошується на операційну локальність виконання бізнес-функціональностей в оптимальній кібер-фізичній інфраструктурі співробітниками, що мають не ролі (позиції), а критичні знання про нові технології, включаючи хмарні федеральні послуги машинного навчання (Federative Machine Learning) [2,3]. Візуалізація структурованих даних, що ієрархічно еволюціонують, із часом перетворюється у проблему. Існує кілька підходів до її вирішення [4]. Такі методи, як анімовані або зіставлені деревоподібні візуалізації, не здатні забезпечити хороший огляд часових рядів і не мають виразності в передачі змін у часі. Вкладені потокові графи забезпечують краще розуміння еволюції даних, але не мають чіткої схеми

ієрархічних структур на заданому часовому кроці. Ці підходи часто обмежуються статичними ієрархіями або виключають складні ієрархічні зміни даних, обмежуючи варіанти їх використання. Пропонований метод дозволяє плавно переходити між деревоподібними картами та вкладеними графами потоків, дозволяючи досліджувати компроміс між динамічною поведінкою та ієрархічною структурою. Оскільки технологія обробляє топологічні зміни всіх типів, вона підходить для широкого загалу додатків. Демонструється корисність методу на декількох прикладах, оцінюючи його за допомогою дослідження користувача, з наданням повного вихідного коду.

У дослідженні [5] розпізнавання образів контрольних карток (CCPR) у більшості попередніх методів використовувався класифікатор для позначення аномальних ККТ. Однак довгострокові дані контрольної картки часто містять велику кількість невеликих аномальних патернів з характеристиками, відмінними від тих, що видно при глобальному перегляді всієї картки. Існує також висока ймовірність того, що локальні аномальні патерни заслуговують на аналіз. Представлено нову схему розпізнавання образів багатомасштабних контрольних діаграм, MS-CCPR, яка не фокусується на класифікації даних з однієї діаграми. Натомість схема використовує пропоноване представлення даних на основі гістограм у поєднанні зі зіставленням під послідовностями часових рядів для виявлення аномальних закономірностей у різних масштабах з довгого ряду контрольних карт. Експериментальні результати демонструють ефективність запропонованої структури ефективному виявленні графічних патернів у різних масштабах, перевершуючи сучасні алгоритми зіставлення послідовностей часових рядів.

У роботі [6] розглянуто проблему розпізнавання образів перешкод (JPR), коли деякі образи не мають навчальних вибірок. Існуючі роботи у цьому напрямі описують розпізнавання лише відомих моделей перешкод і розглядають ситуації із невідомими перешкодами. Автори пропонують схему JPR на основі навчання з нульовим пострілом (ZSL). На першому кроці вводиться контрольований процес навчання для вивчення уявлення прихованих ознак відомих шаблонів перешкод. Потім пропонується неконтрольований підхід до класифікації для розпізнавання різних моделей перешкод. Нарешті, як відомі, і невідомі шаблони перешкод класифікуються у просторі прихованих ознак. Результати моделювання показують, що запропонована схема забезпечує відмінну продуктивність під час роботи з відкритими задачами JPR.

В останні роки розпізнавання [7] виразу обличчя (FER) привернув велику увагу через його широке застосування. Хоча певного прогресу було досягнуто завдяки появі глибокого навчання, проблема, пов'язана зі зміною поз, залишається. Більшість традиційних підходів здебільшого виконують FER у лабораторно-контрольованому середовищі, а FER у дикій природі приділяється щодо меншої уваги. Для реалізації FER у дикій природі можливим рішенням була б модель розпізнавання виразів, інваріантна до пози, але для неї існує проблема нестачі навчальних даних. Достатні навчальні дані з надійними мітками виразів для FER зазвичай недоступні. Ця стаття присвячена вирішенню проблеми моделювання варіації пози на зображеннях обличчя та використання зашумлених даних в Інтернеті для підвищення продуктивності FER. Пропонована модель реалізується наскрізним способом зі слабким контролем і має низку переваг. По-перше, вона використовує масивні зашумлені розмічені дані підвищення продуктивності класифікатора FER, навченого на невеликому наборі чистих міток. По-друге, пропонується нова мережа моделювання поз для адаптивного фіксування невідповідностей у просторі глибоких уявлень зображень обличчя за різних поз голови. Запропонована модель дозволяє вивчити уявлення інваріантних до пози виразів. Для використання надійної інформації в зашумлених даних сформульована мережа моделювання шуму, яка здатна вивчати відображення простору ознак залишки між чистими мітками і зашумленими мітками. Запропонований підхід перевірено на 4-х загальнодоступних тестах FER: AffectNet, RAF-DB, SFEW та BU-3DFE. Великі експерименти показують, що запропонований метод кращий за інші сучасні методи. Іноді розпізнавач сміттєвих даних ефективніше працює, ніж розпізнавач функціонального ряду.

Прогноз фінансових часових рядів [8] є основним об'єктом вивчення у галузі фінансової економетрики. На першому етапі виконується відділення будь-яких систематичних варіацій цих рядів від їх випадкових рухів. Систематичні зміни можуть бути викликані трендами, сезонними та циклічними коливаннями. Економетричні моделі включають різні рівні складності для імітації різноманітних закономірностей. Проте алгоритми машинного навчання можна використовувати прогнозування нелінійних часових рядів, оскільки можуть навчатися і розвиватися разом із фінансовими ринками. Найбільш стандартним економетричним підходом до прогнозування тенденцій фінансових часових лав є методологія Бокса та Дженкінса (1970). Розглянутий підхід заснований на визначенні відповідних систематичних варіацій часового ряду (тренд, сезонні або циклічні ефекти).

3. Постановка завдання

Мета дослідження – підвищення ефективності пошуку та розпізнавання логіко-часових патернів, що семантично утворюють бізнес-функціональності у 8-годинному часовому фреймі скріншотів зі сміттєвими даними.

Початкові дані: файли бізнес-потоків у 8-годинному часовому кадрі, що містять патерни формування бізнес-функціональностей, а також інтервали часу для формування бізнес-функціональностей. Значимість даних визначається часовим інтервалом, а також вручну призначеними компонентами, які можуть мати незначний час виконання.

Задачі:

1) формування моделі – структур даних для опису потоку невизначених множин патернів, сукупність фрагментів, які мають ідентифікатори початку

та кінця. Багато патернів має в кінці кожного фрагмента цифровий ідентифікатор;

2) розробка унітарно кодованої матричної моделі як декартового добутку множини примітивів-символів та set-вектора дискретної послідовності-реалізації заданого часового бізнес фрейму;

3) синтез логіки алгоритмів для пошуку, ідентифікації та класифікації патернів-множин. Визначення прийнятного формату вихідних даних та супровідної статистики;

4) верифікація знайденої сукупності патернів на основі метрики перетину-об'єднання;

5) тестування та верифікація програми, оформлення коду алгоритму та опису структур даних.

4. Основний матеріал дослідження

4.1. Розробка моделі для пошуку та ідентифікації бізнес-патернів. Модель для синтезу логіки алгоритму пошуку та ідентифікації є вихідною або первісною матрицею, форматованою в метриці параметрів <string-primitives – business-flow>, що представлено на рис. 3.

Strings	Patterns											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1				1							
B		1		1								
C			1					1				
D							1					
E					1							
F						1						
G												
H									1			
K											1	
L										1		1

Рис. 3. Матриця у метриці параметрів
(**Fig. 3.** The matrix in the parameter metric)
<string-primitives – business-flow>

Тут кожен стовпець має лише одну одиницю, оскільки скріншот не може (не повинен) містити два активні вікна різних функціональностей. Рядки містять кінцеву кількість одиниць, які формують участь примітиву в одному або декількох патернах в рамках заданого часового фрейму (8 годин). Як правило, послідовність сусідніх стовпців-векторів, що мають одиниці, складають патерн своїми літерними (стринговими) ідентифікаторами. Тривалість одного патерну регулюється часовим інтервалом, який має певну статистику обробки даних. Саме часовий інтервал виконання бізнес-функціональності є верифікатором-розділювачем одного патерну від іншого, якщо в них використовуються однакові множини string-ідентифікаторів. що беруть участь у формуванні бізнес-функціональностей.

Прикладом-результатом виконання цієї процедури є табл. 1. Тут об'єднання підмножин-примітивів виділені різними кольорами, де перетини утворених підмножин по групах рядків і стовпців (знайдені еквівалентні групи, що становлять патерни, табл. 2) дорівнюють порожній множині, а їх об'єднання становить бізнес-потік:

Знайти кореляцію між примітивами, що становлять патерн, шляхом застосування алгоритмів клас{тер, ифік}-ації є сутністю завдання. Для підвищення продуктивності навчання можна використовувати розподілену структуру, представлену на рис. 4.

Таблиця 1 – **Приклад виконання процедури**

Strings	Patterns											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1											
B		1										
C			1									
D						1						
E					1							
F				1								
G							1					
H								1				
K									1			
L												1
M											1	
N										1		

Таблиця 2 – **Виділення патернів**

Strings	Patterns											
	1	2	3	4	5	6	7	8	9	10	11	12
A	1											
B		1										
C			1									
D						1						
E					1							
F				1								
G							1					
H								1				
K									1			
L										1		1
M											1	
N										1		

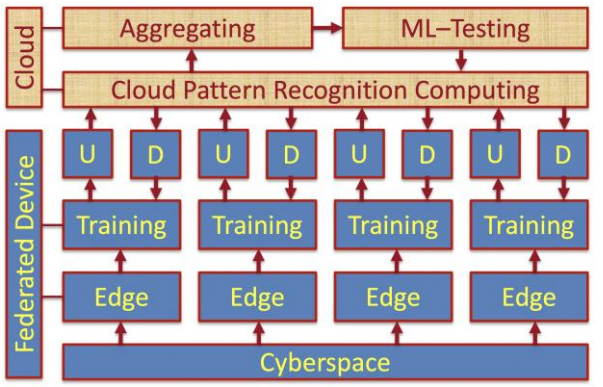

Рис. 4. Масове машинне навчання FML
(**Fig. 4.** FML massive machine learning)

Далі необхідно формувати квадратичну матрицю частотної взаємодії букв-примітивів (граф частотностей переходів на буквах) з метою визначення кореляції між ними та виділення підмножини букв, відповідних патернам. Тут вирішується задача еквівалентування підмножин, які повинні перетинатися, але становити повну множину літер (патернів) за її об'єднанні. Окремий випадок взаємодії патернів на діаграмах та в картах Карно (О. Дрозд) показаний на рис. 5.

Розглядається послідовність фрагментів-примітивів

$$G_i = ID_i\left(F_1, F_2, \ldots F_j, \ldots F_n\right)$$

з одним цифровим ID-ключом, де кожен фрагмент символів-літер – рядок має виражені початок і кінець:

$$F_i = \left(\text{start}, a_1, a_2, \ldots a_j, \ldots a_m, \text{end}-\text{ID}_i\right).$$

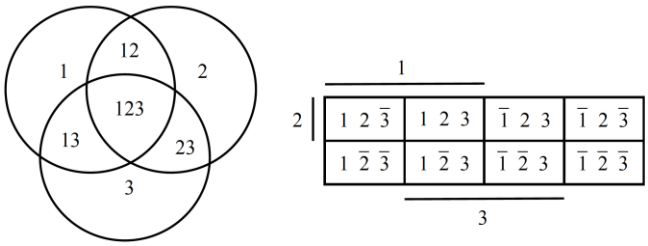

Рис. 5. Взаємодія патернів ID-послідовності, що підлягає поділу (**Fig. 5.** Interaction of ID-sequence patterns to be separated)

Сукупність із n послідовностей дій створює бізнес-потік, 8 годин:

$$G = (G_1, G_2, \ldots G_i, \ldots G_n).$$

Знайти функціонально подібні послідовності та об'єднати їх в один клас означає побудувати set-модель бізнес-процесу, яка наочніше і простіше при аналізі. Сформувати *k* різних класів функціональностей на потоці *G*. Виконати верифікацію *k* знайдених класів еквівалентних послідовностей дій.

Слід пам'ятати, що еквівалентні послідовності можуть відрізнятися друг від друга за подібністю 90 відсотків і менше.

Еквівалентування – процес пошуку та ідентифікації сукупності (підмножини) послідовностей дій (послідовностей функціональних фрагментів) бізнес-процесу, що мають метрично однакові властивості (рефлексивність, симетричність і транзитивність). Set-вектор – позначення сукупності (множини) компонентів у формі впорядкованої послідовності символів чи чисел. Будь-яка множина (підмножина) еквівалентно по відношенню до належності своїх елементів. Дві підмножини еквівалентні, якщо вони мають однакові елементи. У матричній формі підмножина представлена квадратом елементів діагоналі. Нова метрика: відношення між кінцевим числом об'єктів є еквівалентним, якщо циклічна сума відстаней між ними дорівнює нулю:

$$\bigoplus_{i=1,n} d_i = 0.$$

Сильна зв'язність графа є сукупність вершин з відсутністю зв'язків, крім відношення власності. Матриця множини є двовимірною структурою, координати якої усюди визначені одиницями

4.2. Алгоритм синтезу моделі:

1. Вихідні дані: текст-вектори, що мають початок-і кінець-ідентифікатори, що перетворюються на числа-літери set-вектора:

$$S = (S_1, S_2, \ldots S_i, \ldots S_n).$$

2. Формування рядків для бінарної матриці set-векторів на метриці унікальних чисел за правилом: $M_{ij}(S_j) = 1$. В результаті буде сформовано *m* рядків матриці, що дорівнює кількості послідовностей дій або set-векторів. Число стовпців дорівнює кількості унікальних квадратиків-літер-цифр у вектор-універсумі *U*: if Symbol $\notin U$ then $p = p + 1$. Останнім буде *p*-індекс, що дорівнює потужності унікальних квадратиків у всіх послідовностях дій (табл. 3).

Таблиця 3 – **Формування рядків**

S1=ABC
S2=ABC
S3=DEF
S4=KLN
S5=ABC
S6=MPT
S7=KLN
S8=DEF

G=	A	B	C	D	E	F	K	L	N	M	P	T
U=	1	2	3	4	5	6	7	8	9	10	11	12
S1	1	1	1									
S2	1	1	1									
S3				1	1	1						
S4							1	1	1			
S5	1	1	1									
S6										1	1	1
S7							1	1	1			
S8				1	1	1						

Перевпорядкування рядків-послідовностей двійкової матриці призводить завжди до діагональної структури однорідних функціональностей, яка показує обсяг кожного виду діяльності оператора в рамках 8-годинного часового фрейму (табл. 4).

Таблиця 4 – **Перевпорядкування рядків**

G=	A	B	C	D	E	F	K	L	N	M	P	T
U=	1	2	3	4	5	6	7	8	9	10	11	12
S1	1	1	1									
S2	1	1	1									
S5	1	1	1									
S3				1	1	1						
S8				1	1	1						
S4							1	1	1			
S7							1	1	1			
S6										1	1	1

При цьому чітко видно, що перетин діагональних підматриць, що створюють еквівалентні послідовності дій, дорівнює порожній множині, а їх об'єднання – сукупному бізнес-процесу.

3. Виконується квадратичне паралельне xor-порівняння всіх рядків матриці між собою для формування оцінок подібності, зведених у відповідну квадратичну матрицю

$$S(S_i, S_j) = \sum_{i=1}^{n} \left(S_{i \underset{i_i=1,n}{\wedge}} S_j \right) \Big/ \sum_{i=1}^{n} \left(S_{i \underset{i_i=1,n}{\vee}} S_j \right),$$

розмірність якої дорівнює потужності послідовностей дій у бізнес-потоку (табл. 5).

Таблиця 5 – **Результат xor-порівняння**

Sim	S1	S2	S3	S4	S5	S6	S7	S8
S1	1	1,0			1,0			
S2	1,0	1						
S3			1					1,0
S4				1			1,0	
S5	1,0				1			
S6						1		
S7				1,0			1	
S8			1,0					1

4. Еквівалентування послідовностей дій (set-векторів) за класами, що визначаються максимальними оцінками їхньої подібності. Береться перший рядок, в якому вибираються всі координати рівні 1 (або деякому максимальному значенню), які ідентифікують еквівалентний клас послідовностей-стовпців. Потім усі стовпці та рядки, що відповідають еквівалентним послідовностям дій (S_1, S_2, S_5), обнуляються. Можна також викреслювати їх, зменшуючи розмірність матриці на кожному кроці, що показано на табл. 6, 7.

Таблиця 6 – **Еквівалентування set-векторів за класами (початок процесу)**

Sim	S1	S2	S3	S4	S5	S6	S7	S8
S1	1	1,0			1,0			
S2	1,0	1						
S3			1					1,0
S4				1			1,0	
S5	1,0				1			
S6						1		
S7				1,0			1	
S8			1,0					1

Таблиця 7 – **Еквівалентування set-векторів за класами (підсумок)**

Sim	S3	S4	S6	S7	S8
S3	1				1,0
S4		1		1,0	
S6			1		
S7		1,0		1	
S8	1,0				1

Код для пошуку еквівалентних класів послідовностей дій має такий вигляд:

$$M_{ij}; i, j++; \text{if } M_{ij} = 1,0 \text{ then } k++; Ek = j;$$

$$M_i - \text{delete}; M_j - \text{delete}.$$

5. Далі розглядається черговий рядок (після видалення рядків і стовпців завжди розглядається перший рядок і перша координата в ньому), де є значуща координата, що дорівнює 1. Алгоритм триває до повного обнулення (викреслення) всіх координат, яке свідчить про побудову всіх класів еквівалентних послідовностей дій (табл. 8).

Таблиця 8 – **Результат алгоритму**

Sim	S4	S6	S7
S4	1		1,0
S6		1	
S7	1,0		1

Sim	S6
S6	1

6. Отже, найпростіший алгоритм викреслення рядків та стовпців за суттєвими координатами дає можливість розбити матрицю на класи еквівалентностей:

$$E = \{\{S_1, S_2, S_5\}, \{S_3, S_8\}, \{S_4, S_7\}, \{S_6\}\}.$$

Еквівалентування можна проводити і по першій двійковій матриці послідовностей дій, але лише в тому випадку, коли критерієм еквівалентності буде повний збіг пари послідовностей дій по всіх координатах. В іншому випадку необхідно задавати нижню межу подібності (наприклад, >0,8), відносно якої слід вважати послідовності дій, що належать до одного класу еквівалентності.

7. Верифікація отриманого результату шляхом виконання процедур перетину пар еквівалентних класів, яке має бути порожньою множиною. Об'єднання класів має дорівнювати бізнес-потоку.

4.3. Перетворення структури ГСА на множину елементів-функцій.

Задача. Перетворити структуру ГСА на множину елементів-функцій, що підлягають виконанню, інваріантному до порядку. Це означає – покрити сукупність елементів активності оператора та визначити схожість – відмінність кожної послідовності дій, що належить до класифікованого виду функціональності. Інакше, визначити загальне ядро та диференціатори для кожної послідовності дій у вигляді підмножини літер універсуму. Таких класів може бути кінцева множина. Заміна структури множиною означає усунення істотних деталей моделі процесу чи явища. Однак для вирішення великої кількості завдань комп'ютингу структурні особливості створюють обчислювальну складність алгоритму без явних і видимих переваг у кінцевому результаті. Іншими словами, якщо є можливість розв'язати задачу примітивної безадресної (безструктурної) логіки квантового обчислювача (теорії множин), то алгоритми перетворюються на кілька рядків коду, суть яких – поєднання непустих результатів перетину вхідного набору з рядками регулярної (унітарної) таблиці істинності, що в алгебрі логіки диз'юнкція несуперечливих кон'юнкцій. Це є шлях до спрощення структурно складних алгоритмів, що мають багато умов та паралельних шляхів для отримання результату.

Підстава для виконання завдання: теоретико-множинне подання як завгодно складної структурної моделі бізнес-процесу, яка є не тільки компактною, але і більш зрозумілою і доступною для користувача, рис. 6.

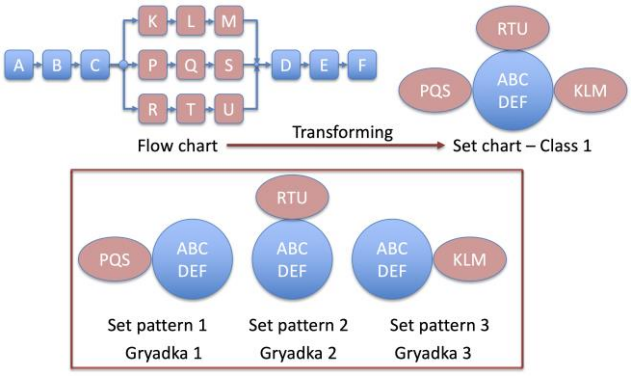

Рис. 6. Трансформування структури у множину (**Fig. 6.** Transforming the structure into a set

Звичайно, форма зображення класів послідовностей дій має значення для користувача з погляду

візуалізації та зручності сприйняття інформації про зміст бізнес-процесу. Один з можливих варіантів зображення двох класів патернів-послідовностей наведений на рис. 7

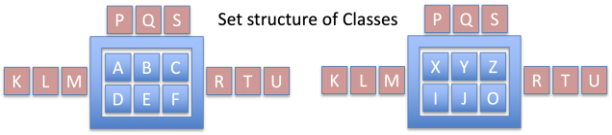

Рис. 7. Set-структури двох класів послідовностей дій
(**Fig. 7.** Set structures of two classes of action sequences)

Вихідна інформація представлена у вигляді гілок flow chart, які створюють послідовності дій (патерни) як відбитки виконання функціональностей операторами. Set-модель послідовності представляється у вигляді підмножини унікальних символів, інваріантних до повторень та порядку прямування. Це дає можливість значно спростити модель бізнес-процесу для подальшого аналізу. Set-послідовність є ідеальною формою визначення кінцевого числа класів функціональностей, і навіть для ідентифікації відмінностей між послідовностями одного класу. Таким чином, послідовність G_i, як примітив-патерн бізнес-процесу має такий формат:

$$G_i = \{S_i, D_i\},$$

де S_i – ядро класу послідовностей дій, яке є загальним атрибутом, складеним із букв-примітивів, D_i – диференціатори послідовностей, які створюють різницю між послідовностями одного класу. Таблиця структури послідовностей дій з доповненнями має вигляд виражених подібностей та відмінностей (табл. 9).

Таблиця 9 – **Таблиця структури послідовностей дій**

G=	A	B	C	D	E	F	K	L	N	M	P	T	1	2	3	4	5	6	7	8
U=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S1	1	1	1										1							
S2	1	1	1											1						
S5	1	1	1												1					
S3				1	1	1										1				
S8				1	1	1											1			
S4							1	1	1									1		
S7							1	1	1										1	
S6										1	1	1								1

Алгоритм класифікації містить такі пункти:

1) вибір чергової послідовності дій для її перетину з рештою інших з метою визначення непустого перетину, яке створює ядро класу;

2) формування різниці між послідовностями одного класу шляхом виконання операції симетричної різниці.

3) інкремент номера чергової послідовності на множині, що залишилася, з метою повторення пунктів 1 і 2 алгоритму для визначення чергового класу. Алгоритм триває до того часу, поки існують ще послідовності, не включені в жодного класу.

Алгоритм по суті містить дві процедури:

1) класифікація всіх рядків-послідовностей шляхом пошуку максимальної подібності (з нижньою межею >0,8) між рядками таблиці істинності;

2) визначення відмінностей-диференціаторів між рядками кожного класу шляхом виконання хог-операцій між послідовностями дій.

Вхідні дані. Вихідний файл містить рядки, що становлять послідовності дій, обмежені маркерами (st, en). Кількість таких послідовностей створює робочий день одного оператора, у разі йдеться про двох операторах, де перший представлений двома днями, а другий – одним. Рядки між маркерами en і st ігноруються. Формат даних рядка представлений інтегрально і роздільно п'ятьма полями-атрибутами, які у сукупності створюють кортеж, який має бути ідентифікований символом (цифрою) універсуму літер, які необхідно і досить формують заданий бізнес-процес. Структура вектору ідентифікаторів бізнес-процесу має такий вигляд:

$$P = (12342318)\ (14567331)\ (1234445635)\ (123455334667)\ (12343234555).$$

При цьому універсум примітивів U=(12345678) представлений вектор-множиною унікальних номерів, яким ставляться у відповідність стрінги-цеглинки. Примітиви створюють вектор ідентифікаторів бізнес-процесу.

Алгоритм створення універсуму примітивів-літер містить такі пункти:

1) перегляд чергового рядка Activity Name. Якщо вона раніше не зустрічалася, вона заноситься до масиву універсуму примітивів U і їй присвоюється черговий унікальний номер в нумерації універсуму;

2) цей номер також присвоюється рядку даних Activity Name, який формує вектор номерів примітивів P, що дорівнює за довжиною числа рядків таблиці бізнес-процесу;

3) якщо рядок раніше зустрічався у таблиці бізнес-функціональності, то йому надається відповідний номер з універсуму примітивів U. Процес нумерації рядків U триває до повної ідентифікації їх номерами примітивів.

Після синтезу універсуму примітивів і вектору ідентифікації P рядків бізнес-процесу виконується алгоритм формування двійкової матриці послідовностей дій, у метриці універсуму примітивів шляхом використання вектору P: виконується занесення до початкового рядку матриці одиничних значень за тими координатами, які фігурують у векторній формі дій $M(i,j) = P(k)$, що регламентується ідентифікаторами початку та кінця чергової послідовності. Число послідовностей n формує кількість рядків матриці. Після синтезу матриці **визначаються класи еквівалентних послідовностей дій** шляхом виявлення подібності між усіма парами послідовностей. При цьому всередині одного класу формуються диференціатори, що розрізняють кожну послідовність усередині класу функціональними особливостями.

Уточнення. Необхідно враховувати час створення символів-рядків у кожному фрагменті чи послідовності. Матриця визначена в метриці універсуму примітивів-символів-рядків, декартово помноженої на кількість послідовностей, кожна з яких містить кінцеву кількість фрагментів. При цьому координата матриці визначена чисельним значенням часу, який

інтегрально було витрачено на формування символ-рядку у масштабах послідовності. Приклад матриці «універсум-послідовності» наведений у табл. 10.

Таблиця 10 – **Приклад матриці «універсум-послідовності»**

G=	A	B	C	D	E	F	K	L	N	M	P	T	1	2	3	4	5	6	7	8
U=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S1	2	4	7										5							
S2	4	3	2											5						
S5	4	5	5												6					
S3				3	5	4										7				
S8				2	5	7											4			
S4							6	7	7									3		
S7							5	4	3										6	
S6										5	3	7								3

Тут значення кожної координати визначається сумою часів формування літери-рядка у всіх фрагментах однієї послідовності

$$M_{ij} = \sum_{j=1}^{n} a_j, \quad a_j \in G_i.$$

Метрика для підрахунку подібності між рядками матриці використовує трансформацію формули подібності для двійкової матриці:

$$S\left(S_i, S_j\right) = \frac{\sum_{i=1}^{n}\left(S_i \underset{i_i=1,n}{\wedge} S_j\right)}{\sum_{i=1}^{n}\left(S_i \underset{i_i=1,n}{\vee} S_j\right)} \approx \frac{\sum_{i=1}^{n}\left[\min_{i=1,n}\left(S_i, S_j\right)\right]}{\sum_{i=1}^{n}\left[\max_{i=1,n}\left(S_i, S_j\right)\right]}.$$

Наприклад, подібність між першим і другим рядками матриці визначається nfrimи обчисленнями:

$$S\left(S_i, S_j\right) \approx$$

$$\approx \frac{\sum_{i=1}^{n}\left[\min_{i,j=1,n}(2,4),(4,3),(7,2),(5,0),(0,5)\right]}{\sum_{i=1}^{n}\left[\max_{i,j=1,n}(2,4),(4,3),(7,2),(5,0),(0,5)\right]} =$$

$$= \frac{\sum_{i=1}^{n}\left[\min_{i,j=1,n}(2),(3),(2),(0),(0)\right]}{\sum_{i=1}^{n}\left[\max_{i,j=1,n}(4),(4),(7),(5),(5)\right]} = \frac{7}{25} = 0,28.$$

Якщо символ-рядок, що ідентифікується літерою, більш важлива для формування послідовності дій, навіть за малого значення часу, то їй надається максимальна вага в масштабі всієї матриці бізнес-послідовностей.

4.4. Макро-алгоритм визначення класів еквівалентностей:

1) Визначення універсум-алфавіту для формування стовпців матриці бізнес-процесу. Для цього проглядається весь потік даних з метою виділення лише множини оригінальних символів-рядків.

2) Фаза заповнення чергового (поточного) рядка матриці шляхом перебору символів у фрагментах однієї послідовності дій для адитивного формування часових характеристик кожного символу, що бере участь в інтегральній функціональності послідовності:

$$M_{ij} = M_{ij} + t \cdot \left(G_k\right) \leftarrow G_k = a_j,$$

де символ-рядок $a_j \in U$ створює універсум і одночасно належить послідовності $a_j \in G_i$, що формує j-координату в i-рядку матриці.

3) Фаза формування квадратичної матриці подібності шляхом застосування мінімальної метрики для визначення Similarity.

4) Фаза визначення класів еквівалентних послідовностей шляхом пошуку максимальних значень подібності між рядками матриці бізнес-процесу.

5) Фаза перевірки або верифікації отриманих класів еквівалентностей шляхом їх перетину-об'єднання, що має бути порожньою множиною та бізнес-потоку відповідно.

Програмна реалізація дозволила отримати 100 послідовностей та 2000 примітивних символів, де є порожні за часом символи. Необхідно мінімізувати універсум літер шляхом виключення незначних для класифікації символів.

Алгоритм визначення кластерів у вигляді підмножини послідовностей дій з універсуму примітивів на основі аналізу потоку даних, використовуючи зрізане число атрибутів.

1) Синтезувати квадратичну матрицю подібності послідовностей процесів (рис. 8).

2) Прибрати символи X, потім повторити побудову матриці подібності. Склеювати повторювані цифрові ідентифікатори послідовностей дій.

3) Розділити потік днями та операторами. Повторити побудову матриць подібності.

4) Розглянути можливість суттєвого зменшення символів алфавіту шляхом видалення праворуч несуттєвих диференціаторів.

5) Повторити побудову матриць подібності на мінімальному множині символів.

0.0	0.13400960828945000	0.04917119487354380	0.06820125433918990	0.04533042877744030	0.13865343254882600	0.005837307927746210	0.05538760660993240	0.0
0.0	0.0	0.16021377030235100	0.04928536780545550	0.16842554680827900	0.20038765470381400	0.015956796302356700	0.13396072159004400	0.005540510174220970
0.0	0.0	0.0	0.07440620438881340	0.17489600686236400	0.0951858804499107	0.04411281858847900	0.29195646562776400	0.001697300371896250
0.0	0.0	0.0	0.0	0.10142093439855100	0.05863023054153540	0.011140246276971300	0.03231413021384780	0.01477639357761750
0.0	0.0	0.0	0.0	0.0	0.22130789859102500	0.004697259681964200	0.24044312439889100	0.005585129042233100
0.0	0.0	0.0	0.0	0.0	0.0	0.006200139457040040	0.137137855148924	0.006871205913370990
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.016329516212738700	0.002352471009560550
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0028112634885094200
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Рис. 8. Матриця подібності послідовностей дій (**Fig. 8.** Matrix of similarity of action sequences)

Визначення подібності за часом дає 14 кластерів:

[[0, 1, 5, 10, 18, 24, 25, 27, 28, 29, 30, 31, 34, 35, 36, 37, 38, 43, 44, 48, 49, 54, 56, 57, 61, 62, 63, 64, 65, 69, 71, 72, 74, 75, 76, 81, 82, 84, 86, 87, 88, 90, 93, 100], [], [2, 4, 7 , 9, 13, 16, 19, 20, 21, 22, 23, 55, 60, 70, 77, 78, 80, 85], [3, 11, 12, 33, 42, 45, 52, 59, 66 , 67, 68, 79, 83, 89, 91, 94, 99], [], [], [6], [], [8], [], [], [], [], [], [14, 15, 32, 41, 53, 92], [], [], [17, 95, 96, 97, 98], [], [], [], [], [], [], [], [], [26], [], [], [], [], [], [], [], [], [], [], [], [], [39], [40], [], [], [], [], [], [46], [47], [], [], [50], [51], [], [], [], [], [], [], [58], [], [], [], [], [], [], [], [], [], [], [], [], [ ], [], [73], [], [], [], [], [], [], [], [], [], [], [], [], [] , [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], []]

Перевірити кількість послідовностей дій. Верифікувати по потоку подібність послідовностей дій, які потрапили у кластери. Інтегральний час кожної послідовності може допомогти ідентифікувати кластери. Якщо не враховувати час виконання кожного символу, оцінка подібності послідовностей істотно підвищиться. Спробувати визначити схожість по двійковій матриці послідовностей дій. Потім верифікувати результат за часом виконання послідовності. Зменшити кількість атрибутів до двох (прибираються хвости, зменшується потужність алфавіту символів) та порахувати кількість кластерів. У цьому також зростає оцінка подібності.

Алгоритм визначення кластерів у вигляді підмножини букв з універсуму примітивів на основі аналізу потоку даних з використанням усіченої кількість атрибутів. Створюється граф кореляції всіх пар букв із універсуму примітивів, які є сусідніми у вибраному вікні бізнес-потоку. Але сусідніми можуть бути літери, що належать різним функціональностям. Формується квадратична матриця частотностей (але не подібності) сусідніх букв універсуму на потоці даних. При цьому може бути не врахована взаємодія літер А та Б у послідовності А…Б, якщо літери виходять за межі вікна, що є помилкою. Далі використовується метод спектрального аналізу квадратичної матриці для ідентифікації кластерів, в яких неявно заховані послідовності дій (патерни). Перетин кластерів між собою дорівнює порожній множині. Можливо, це найкращий метод для кластеризації бізнес-функціональностей, побудованих з елементів-літер, що створюють бізнес-потік робочого дня. Наразі бібліотека має 350 символів, на яких визначено 331 кластер.

В ідеалі два результати за двома методами (рис. 9) повинні співпадати за підмножиною букв у кластері.

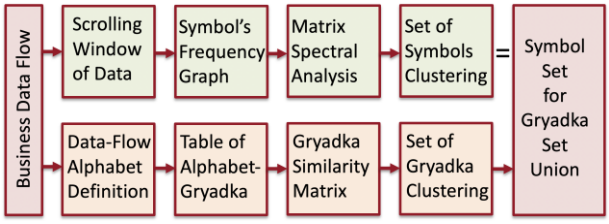

Рис. 9. Подібність та відмінність двох методів кластеризації (**Fig. 9.** Similarities and differences of two clustering methods)

Тут на виході алгоритму в першому випадку формується сукупність підмножин, що не перетинаються, букв-символів, які відповідають основним функціональностям, зашифрованим в бізнес-потоці даних.

У другому випадку виконується кластеризація послідовностей або бізнес-патернів, де всередині кожного кластера знаходяться подібні за функціональностями підмножини літер, які їх формують.

Якщо замінити часові параметри алфавіту на двійкові символи, квадратична матриця подібності послідовностей дій між собою істотно збільшиться. В результаті аналізу матриці подібності були отримані наступні 44 кластери на повній множині послідовностей дій, число яких дорівнює 100, за відсутності видимої кореляції часу виконання послідовностей в кожному кластері.

Необхідно порахувати інтегральний час за кожною послідовністю, щоб порівняти їхню кореляцію в кожному кластері. Час повинен бути приблизно однаковим для всіх послідовностей в одному кластері.

Завдання. Прибрати Idle, та інші несуттєві компоненти, зменшити алфавіт букв за рахунок включення лише 3 атрибутів.

Призначити ваги кожному з атрибутів, що створюють алфавіт, відповідно до ієрархії, ліворуч: $2^4, 2^3, 2^2, 2^1, 2^0$. Якщо символ-атрибут стоїть на першій позиції, йому присвоюється максимальна вага на п'ятій – мінімальна.

Обчислення подібності між собою всіх послідовностей дій потоку даних здійснюється за такою формулою:

$$S\left(S_i, S_j\right) = \frac{\sum_{i=1}^{n}\left[\min_{i=1,n}\left(S_i, S_j\right)\right]}{\sum_{i=1}^{n}\left[\max_{i=1,n}\left(S_i, S_j\right)\right]}.$$

Визначення кластерів на отриманому алфавіті з урахуванням ваги. Створити матрицю частотностей обмеженого числа знаків, а далі побудувати матрицю подібності. Це теж має спрацювати, оскільки ліві атрибути з'являються частіше, ніж праві, що дає підстави вважати частотність метрикою для рівнів ієрархії, де що вище частотність, то вище значення і вище рівень ієрархічності в послідовності дій.

Координата j матриці має інкремент, якщо символ G_{ir} із сукупності атрибутів послідовності дорівнює букві алфавіту A_i на універсумі визначених примітивів:

$$M_{ij} = M_{ij} + 1 \leftarrow G_{ir} = A_j,$$

Перевірити інтегральний час послідовності дій без ідентифікаторів.

Таблиця кореляції векторів примітивів даних у метриці норми (0–1) має вигляд, наведений у табл. 11. Використовується бінарна матриця слів, що формують послідовності дій.

Загальна кількість рядків у потоці даних – 19939. Число послідовностей дій – 100. Алфавіт слів = [176].

Таблиця 11 – **Таблиця кореляції векторів примітивів даних у метриці норми (0–1)**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1.000	0.737	0.500	0.476	0.385	0.370	0.186	0.500	0.105	0.500	0.609	0.444	0.318	0.464	0.364	0.263	0.500	0.375	0.158	0.357	0.565
2	0.737	1.000	0.571	0.579	0.522	0.500	0.188	0.640	0.176	0.640	0.800	0.462	0.273	0.600	0.450	0.353	0.577	0.391	0.167	0.480	0.750
3	0.500	0.571	1.000	0.448	0.567	0.548	0.306	0.559	0.143	0.606	0.600	0.429	0.290	0.733	0.414	0.250	0.606	0.333	0.138	0.531	0.516
4	0.476	0.579	0.448	1.000	0.571	0.545	0.176	0.500	0.286	0.500	0.619	0.440	0.444	0.520	0.687	0.500	0.444	0.500	0.187	0.591	0.500
5	0.385	0.522	0.567	0.571	1.000	0.773	0.197	0.517	0.211	0.517	0.560	0.323	0.348	0.593	0.455	0.300	0.630	0.400	0.200	0.739	0.520
6	0.370	0.500	0.548	0.545	0.773	1.000	0.211	0.552	0.200	0.500	0.538	0.312	0.333	0.571	0.435	0.286	0.552	0.385	0.190	0.708	0.444
7	0.186	0.188	0.306	0.176	0.197	0.211	1.000	0.300	0.061	0.247	0.211	0.313	0.164	0.268	0.197	0.106	0.282	0.206	0.060	0.208	0.197
8	0.500	0.640	0.559	0.500	0.517	0.552	0.300	1.000	0.160	0.724	0.667	0.469	0.276	0.581	0.462	0.280	0.667	0.367	0.154	0.484	0.571
9	0.105	0.176	0.143	0.286	0.211	0.200	0.061	0.160	1.000	0.160	0.200	0.130	0.231	0.167	0.308	0.571	0.115	0.111	0.286	0.190	0.150
10	0.500	0.640	0.606	0.500	0.517	0.500	0.247	0.724	0.160	1.000	0.731	0.516	0.276	0.581	0.462	0.280	0.562	0.367	0.154	0.484	0.571
11	0.609	0.800	0.600	0.619	0.560	0.538	0.211	0.667	0.200	0.731	1.000	0.500	0.333	0.630	0.500	0.350	0.607	0.385	0.136	0.518	0.625
12	0.444	0.462	0.429	0.440	0.323	0.312	0.313	0.469	0.130	0.516	0.500	1.000	0.360	0.484	0.458	0.261	0.424	0.462	0.125	0.303	0.414
13	0.318	0.273	0.290	0.444	0.348	0.333	0.164	0.276	0.231	0.276	0.333	0.360	1.000	0.333	0.471	0.267	0.276	0.474	0.133	0.320	0.240
14	0.464	0.600	0.733	0.520	0.593	0.571	0.268	0.581	0.167	0.581	0.630	0.484	0.333	1.000	0.480	0.292	0.633	0.333	0.160	0.552	0.593
15	0.364	0.450	0.414	0.687	0.455	0.435	0.197	0.462	0.308	0.462	0.500	0.458	0.471	0.480	1.000	0.538	0.357	0.450	0.200	0.417	0.391
16	0.263	0.353	0.250	0.500	0.300	0.286	0.106	0.280	0.571	0.280	0.350	0.261	0.267	0.292	0.538	1.000	0.231	0.211	0.333	0.273	0.300
17	0.500	0.577	0.606	0.444	0.630	0.552	0.282	0.667	0.115	0.562	0.607	0.424	0.276	0.633	0.357	0.231	1.000	0.323	0.154	0.533	0.571
18	0.375	0.391	0.333	0.500	0.400	0.385	0.206	0.367	0.111	0.367	0.385	0.462	0.474	0.333	0.450	0.211	0.323	1.000	0.167	0.423	0.296
19	0.158	0.167	0.138	0.187	0.200	0.190	0.060	0.154	0.286	0.154	0.136	0.125	0.133	0.160	0.200	0.333	0.154	0.167	1.000	0.182	0.200
20	0.357	0.480	0.531	0.591	0.739	0.708	0.208	0.484	0.190	0.484	0.518	0.303	0.320	0.552	0.417	0.273	0.533	0.423	0.182	1.000	0.481
21	0.565	0.750	0.516	0.500	0.520	0.444	0.197	0.571	0.150	0.571	0.625	0.414	0.240	0.593	0.391	0.300	0.571	0.296	0.200	0.481	1.000
22	0.560	0.727	0.613	0.500	0.518	0.500	0.239	0.621	0.182	0.621	0.680	0.517	0.259	0.586	0.400	0.318	0.567	0.357	0.174	0.536	0.708
23	0.440	0.591	0.567	0.737	0.727	0.696	0.214	0.571	0.211	0.571	0.625	0.367	0.348	0.593	0.524	0.368	0.571	0.458	0.200	0.739	0.520
24	0.500	0.640	0.606	0.560	0.517	0.667	0.247	0.667	0.160	0.613	0.667	0.424	0.276	0.581	0.407	0.280	0.562	0.414	0.154	0.533	0.571

Classes=45 (sim>0.6)
[[0, 1, 10, 35, 43, 56, 58, 69], [], [2, 9, 13, 16, 21, 23, 60, 78], [3, 14, 22, 38, 59 , 66, 76, 80, 81, 85, 89, 91, 92], [4, 5, 19], [], [6], [7, 77], [8], [], [], [11], [12, 46], [], [], [15, 32, 45, 51], [], [17], [18], [], [20], [], [], [ ], [24, 48], [25, 42, 53, 61], [26, 27, 29, 30, 36], [], [28, 37], [], [], [31, 55] , [], [33, 34, 54], [], [], [], [], [], [39, 79], [40], [41, 47], [], [], [44], [], [], [], [], [49], [50], [], [52, 71], [], [], [], [], [57], [] , [], [], [], [62, 87], [63], [64], [65], [], [67, 68], [], [], [70, 72], [ ], [], [73], [74], [75, 94], [], [], [], [], [], [], [82], [83], [84], [ ], [86], [], [88], [], [90], [], [], [93, 95, 96, 97, 98], [], [], [], [], [], [99], [100]]

Classes=28 (sim>0.5)
[[0, 1, 10, 20, 21, 30, 34, 35, 36, 38, 43, 44, 56, 58, 60, 61, 69, 71, 72, 76, 77, 84, 88], [ ], [2, 4, 5, 7, 9, 13, 16, 19, 22, 23, 40, 78], [3, 14, 25, 41, 46, 47, 48, 59, 66, 67, 68 , 75, 79, 80, 81, 85, 89, 91, 92], [], [], [6], [], [8, 15], [], [], [11, 42, 90] , [12], [], [], [], [], [17], [18], [], [], [], [], [], [24, 28, 29, 37], [], [26, 27, 33, 87], [], [], [], [], [31, 54, 55], [32, 45, 51, 53], [], [], [ ], [], [], [], [39], [], [], [], [], [], [], [], [], [], [49, 93, 95, 96 , 97, 98], [50], [], [52], [], [], [], [], [57], [], [], [], [], [62], [63, 64], [], [65],[], [], [], [], [70, 100], [], [], [73], [74], [], [], [], [], [], [], [], [82, 83], [], [], [], [86], [], [], [], [], [], [ ], [], [94], [], [], [], [], [99], []]

Виконати верифікацію розбиття шляхом перевірки схожості _en рядків кожної послідовності дій, включеної до кластера. Кластери, складені за _en рядками, є точне рішення, свого роду специфікація. Тому накладання двох квадратичних матриць подібності дасть можливість визначити якість розбиття шляхом обчислення інтегральної оцінки подібності за всіма відповідними координатами двох матриць:

$$S\left(S_i,S_j\right)=\frac{\sum_{i,j=1}^{n}\left[\min\limits_{i=1,n}\left(M_{ij},E_{ij}\right)\right]}{\sum_{i,j=1}^{n}\left[\max\limits_{i=1,n}\left(M_{ij},E_{ij}\right)\right]}.$$

Приведення матриць до двійкового вигляду використовує нижню межу подібності, яка формується евристично: min={0,4 ∨ 0,5 ∨ 0,6}:

$$\text{if}\left(M_{ij}>\min\right)\cdot\left(M_{ij}=1\right)\text{else}\left(M_{ij}=0\right),$$

$$\text{if}\left(E_{ij}>\min\right)\cdot\left(E_{ij}=1\right)\text{else}\left(E_{ij}=0\right).$$

У двійковому вигляді оцінка подібності є більш адекватною та контрастною, яка формує подібність обох матриць:

$$S\left(M,E\right)=\frac{\sum_{i,j=1}^{n}\left(M_{ij}\wedge E_{ij}\right)}{\sum_{i,j=1}^{n}\left(M_{ij}\vee E_{ij}\right)}.$$

У цьому випадку результуюча S-матриця подібності також матиме двійковий вигляд, а інтегральний критерій S(M, E) показуватиме точність кластеризації послідовностей дій, яка приведена до специфікації, що створюється _en рядками. Обчислення подібності через різницю між двома матрицями:

$$S\left(M,E\right)=1-D\left(M,E\right)=1-$$
$$-\sum_{i,j=1}^{n}\left(M_{ij}\oplus E_{ij}\right)\Big/\sum_{i,j=1}^{n}\left(M_{ij}\vee E_{ij}\right)=1-\frac{15}{64}\approx 0,77$$

представлені таблицями рис. 10.

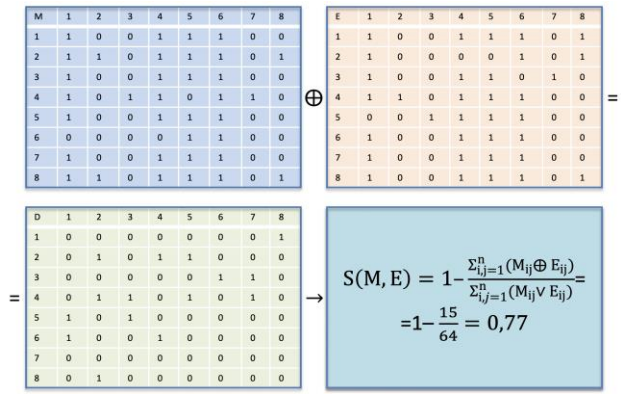

Рис. 10. Обчислення подібності
(**Fig. 10.** Similarity calculation)

5. Висновки та перспективи

Запропоновано та реалізовано метод пошуку та розпізнавання патернів на основі векторного завдання слів, речень та інших символьних послідовностей, які ідентифікують бізнес-патерни у потоках великих даних.

Метод використовує унітарне кодування інформаційних примітивів та даних, які далі подаються у вигляді послідовності або двійкового вектора, який описує бізнес-патерн або функціональність, виконану оператором під час робочого дня.

Структури даних, засновані на унітарному кодуванні інформації, можуть бути оброблені в паралельному режимі за допомогою основних логічних операцій, що формують подібності та відмінності патерн-векторів, чим досягається висока продуктивність обробки великих даних з метою аналізу робочого дня оператора.

Практична реалізація векторного методу дозволила створити програму розпізнавання патернів у потоці великих даних з ймовірністю з ймовірністю 0,77%.

REFERENCES

1. (2021), *Gartner Top 6 Trends Impacting Infrastructure & Operations in 2021*, available at: https://www.gartner.com/smarterwithgartner/gartner-top-6-trends-impacting-infrastructure-operations-in-2021.
2. Joshi A. (2020), *Machine Learning and Artificial Intelligence*, Springer Nature Switzerland AG, 261 p., doi: https://doi.org/10.1007/978-3-030-26622-6.
3. (2021), *IEEE Guide для Архітектурної Framework та Application of Federated Machine Learning*, IEEE Std 3652.1-2020, 69 p., , available at: https://lib.ugent.be/catalog/ebk01:5590000000440557.
4. Bolte F., Nourani, M., Ragan, E. and Bruckner, S. (2020), "SplitStreams: A Visual Metaphor for Evolving Hierarchies", *IEEE Trans. on Vis. & Computer Graphics*, vol. 27, no. 08, pp. 3571-3584, doi: https://doi.org/10.1109/TVCG.2020.2973564.
5. Huang, J.-W., Lee, P.-J. and Jaysawal, B.P. (2022), "Multiscale Control Chart Pattern Recognition Using Histogram-Based Representation of Value and Zero-Crossing Rate", *IEEE Transactions on Industrial Electronics*, vol. 69, no. 1, pp. 684-693, Jan. 2022, doi: https://doi.org/10.1109/TIE.2021.3050355.
6. Han, H., Li, W., Feng, Z., Fang, G., Xu, Y. and Xu, Y. (2021), "Proceed From Known to Unknown: Jamming Pattern Recognition Under Open-Set Setting", *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 693-697, April 2022, doi: https://doi.org/10.1109/LWC.2021.3140145.
7. Zhang, F., Xu, M. amd Xu, C. (2022), "Weakly-Supervised Facial Expression Recognition в Wild with Noisy Data", *IEEE Transactions on Multimedia*, vol. 24, pp. 1800-1814, doi: https://doi.org/10.1109/TMM.2021.3072786.
8. (2022), "Pattern Recognition", Schintler L.A., McNeely CL (eds), *Encyclopedia of Big Data*, Springer, Cham, doi: https://doi.org/10.1007/978-3-319-32010-6_300166.

Received (Надійшла) 21.04.2022
Accepted for publication (Прийнята до друку) 29.06.2022

ВІДОМОСТІ ПРО АВТОРКУ / ABOUT THE AUTHOR

Хаханова Ганна Володимирівна – кандидат технічних наук, доцент, доцент кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Харків, Україна;
Hanna Khakhanova – Candidate of Technical Sciences, associate professor, associate professor of Computer Aided Design of Computers Department, Kharkiv National University of RadioElectronics, Kharkiv, Ukraine;
e-mail: anna.hahanova@nure.ua; ORCID ID: http://orcid.org/0000-0002-1318-7973.

A vector method for finding sequences in big data

Hanna Khakhanova

Annotation. A technological software solution is proposed for metric search and identification of logical-temporal patterns of a business data flow by creating additional vector data structures and a parallel method for their processing. **The subject of research** is the methods of searching and identifying logical-temporal patterns in big data. **The purpose of the study** is to increase the efficiency of searching and recognizing logical-temporal patterns that semantically form business functionality in an 8-hour frame of screenshots with "garbage" data. **Applied methods**: apparatus of set theory and Boolean algebra, metric models for determining parameters for sets of binary vectors, elements of probability theory, theory of algorithms, software modeling. **The results obtained**: a method for searching and recognizing patterns based on a vector problem of character sequences that identify patterns in big data streams using unitary coding of information primitives and data; vector models are unitary-encoded data structures for describing a big data flow as Cartesian products of a set of primitive-string-markers and a discrete sequence of implementation of a given time frame. **The practical significance of the work**: the implementation of the vector method, which made it possible to create a pattern recognition program in a big data stream with a probability of 0.77%.

Keywords: logical-temporal patterns; business functionality; time frame; search and identification; union intersection metric; code verification.